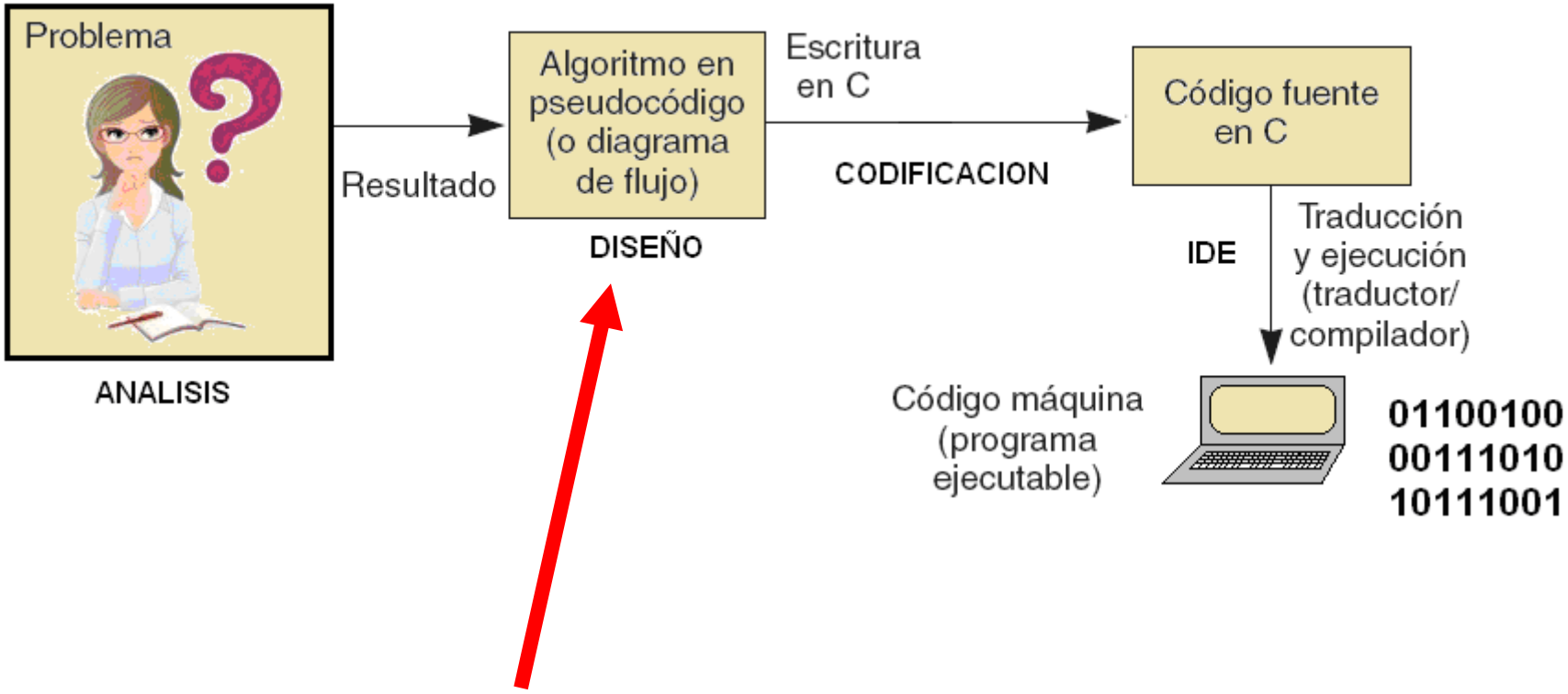
DISEÑO DE ALGORITMOS



Ciclo de V ida del Software:

El proceso de resolución de un problema con una computadora conduce a la escritura de un programa y a su ejecución en la misma. Las fases de resolución de un problema con computadora ó ***ciclo de vida del software*** son:

- **Análisis del problema**
- **Diseño del algoritmo**
- **Codificación**
- **Compilación y ejecución**
- **Verificación**
- **Mantenimiento**
- **Documentación**

ALGORITMO:

(Definición de la Real Academia)

"*Conjunto ordenado y finito de operaciones que permiten resolver un problema*"

- *independiente* del lenguaje de programación.
- *independiente* del ordenador que los ejecuta.

Características:

- ***Precisión***
- ***Repetitividad***
- ***Finitud***
- ***Validez***
- ***Eficiencia***

Partes fundamentales de un algoritmo:

- ***Entrada***, información dada al algoritmo.
- ***Procesamiento***, cálculos necesarios para encontrar la solución del problema.
- ***Salida***, resultados finales de los cálculos.

Análisis del Problema:

- **Comprender correctamente el problema.**
- **¿Datos de Entrada?**
- **¿Datos Salida?**

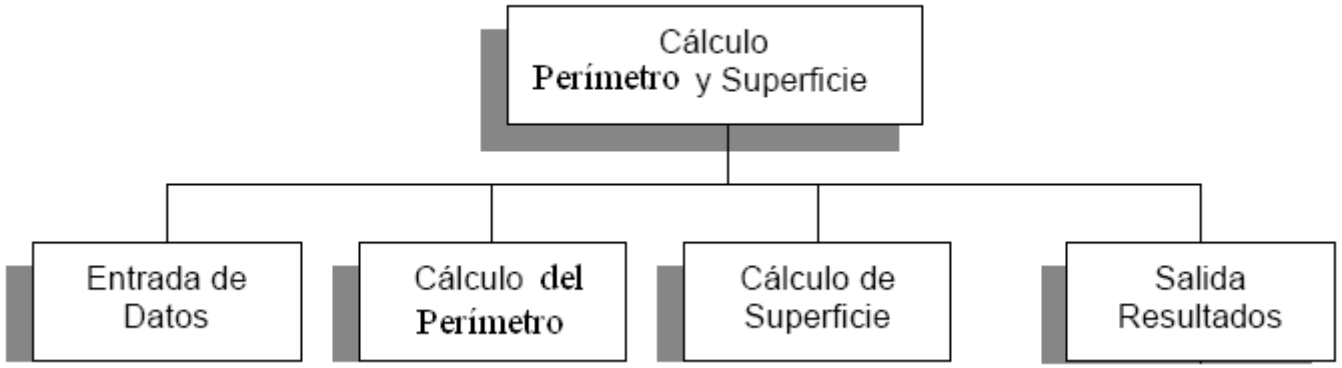
Diseño de Algoritmos:

Vamos a utilizar dos técnicas de programación que no se usan por separado, sino que son complementarias:

- **Programación modular:** Consiste en dividir el programa en partes llamadas módulos, e implementar cada uno de esos módulos por separado.

- **Programación estructurada:** Hace más legible y lógica la estructura del programa utilizando para ello solamente tres tipos de estructuras: selectivas, secuenciales y repetitivas.

PROGRAMACIÓN MODULAR
(Divide y vencerás)



Ejemplo: Calcular la longitud y la superficie de un círculo dado su radio.

Este problema se puede dividir en cuatro módulos:
1) Lectura, desde el teclado, de los datos necesarios.
2) Cálculo del perímetro.
3) Cálculo de la superficie.
4) Mostrar los resultados por pantalla.

PROGRAMACIÓN ESTRUCTURADA

Estructuras de Control:

- **Secuenciales**
- **Alternativas ó Selectivas:**
  - Simples
  - Dobles
  - Múltiples
- **Repetitivas:**
  - Hacer Mientras
  - Para
  - Mientras

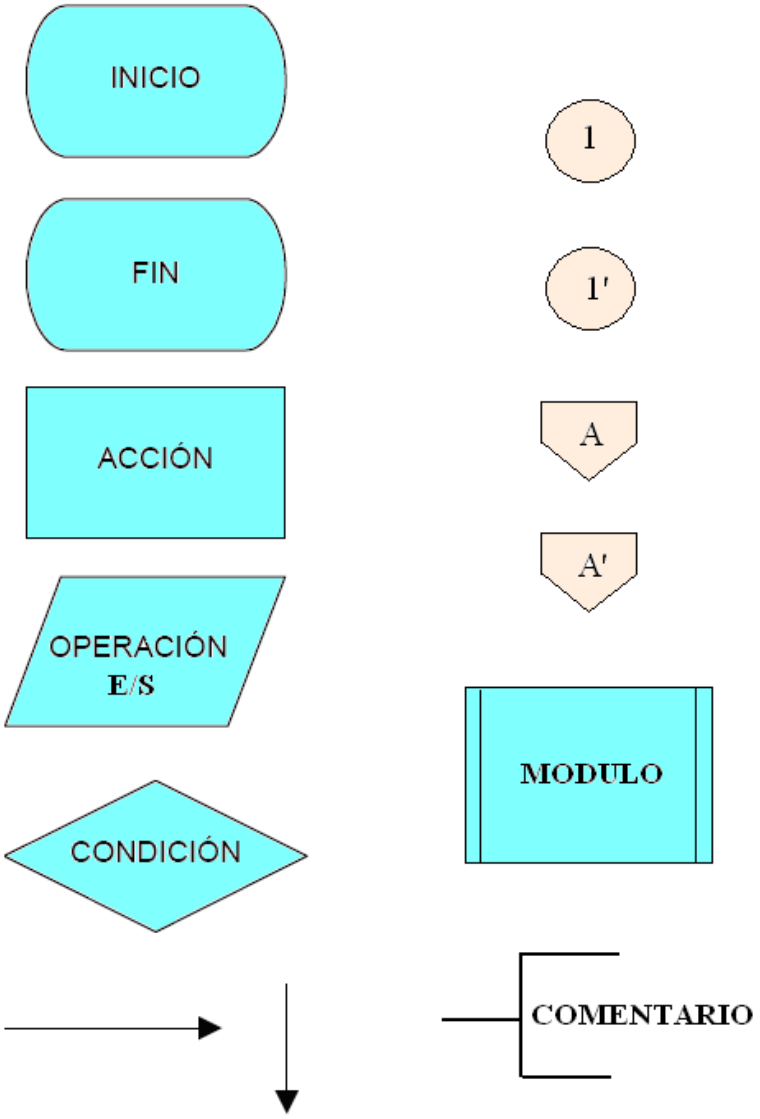
HERRAMIENTAS PARA LA REPRESENTACIÓN DE ALGORITMOS

DIAGRAMAS DE FLUJO

- **Muestran de forma clara el flujo lógico del algoritmo.**
- **Utiliza símbolos gráficos normalizados por ANSI e ISO.**
- **Útil para problemas sencillos.**
- **Difíciles de actualizar.**
- **Los símbolos gráficos de comienzo y fin deberán aparecer una única vez.**
- **El flujo de las operaciones será de arriba a abajo y de izquierda a derecha**

ANSI (*American Nacional Standard Institute*)
ISO (*International Standard Organization*).

Una herramienta libre disponible: Freedfd


INICIO
FIN
ACCIÓN
OPERACIÓN
E/S
CONDICIÓN
1
1'
A
A'
MODULO
COMENTARIO
DISCO
IMPRESORA
TECLADO
PANTALLA

PSEUDOCÓDIGO

- **Permite una aproximación del algoritmo al lenguaje natural.**
- **Permite una redacción rápida.**
- **Podemos centrarnos sobre la lógica del problema olvidándonos de la sintaxis de un lenguaje concreto.**
- **Es fácil modificar el algoritmo descrito.**
- **Es fácil traducir directamente a cualquier lenguaje de programación.**

El ***flujo de control*** del algoritmo, es el orden temporal en el cual se ejecutan los pasos individuales del algoritmo.

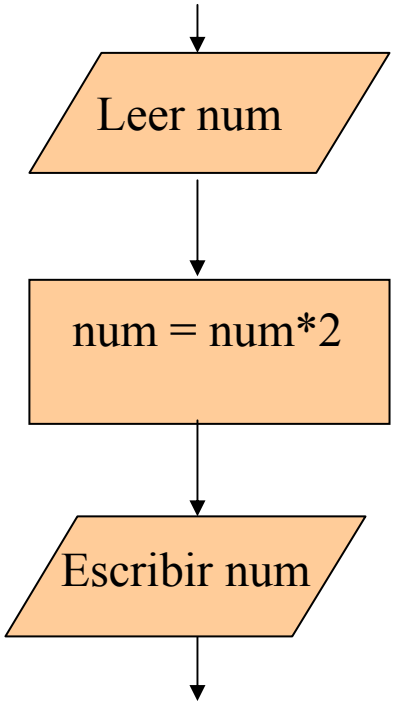
El flujo puede ser según la estructura de control utilizada:

Lineal ó secuencial: un paso a continuación de otro.
Repetitivo: repite automáticamente un grupo de pasos.
Selectivo: selecciona una acción de entre un par de alternativas específicas, según condición.

Inicio y Fin: Por donde empieza y acaba el algoritmo.

ESTRUCTURA SECUENCIAL

DIAGRAMA DE FLUJO



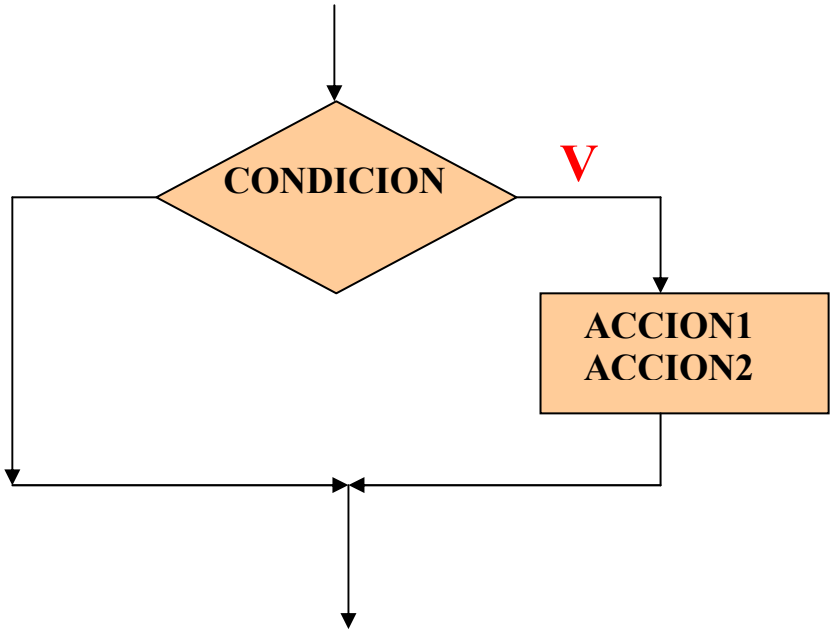
PSEUDOCÓDIGO

```
.....
Leer num
num = num*2
Escribir num
.....
```

ESTRUCTURAS SELECTIVAS

- **Simple:**

DIAGRAMA DE FLUJO

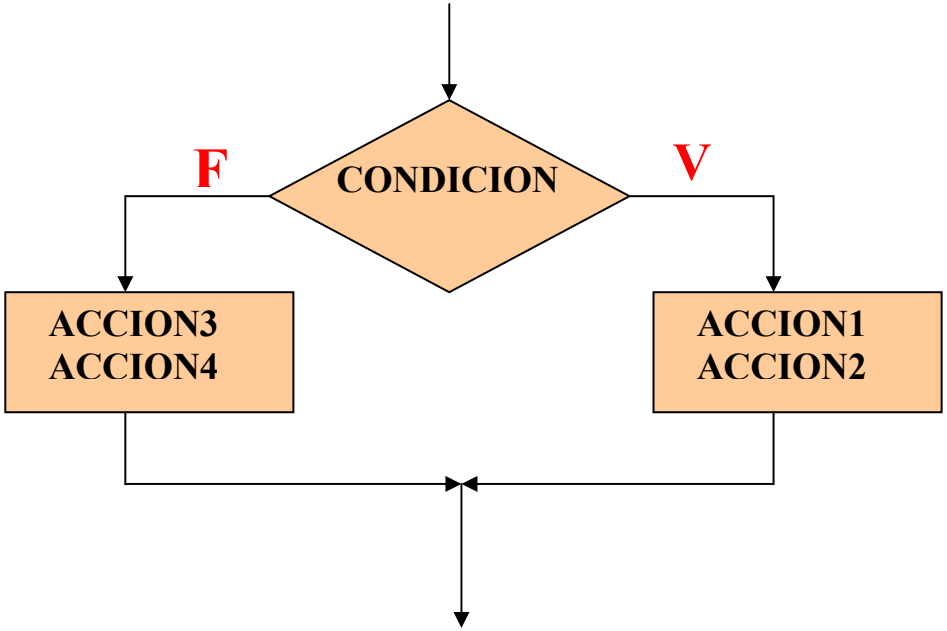


PSEUDOCÓDIGO

```
……….
Sí (condicion)
    accion1
    accion2
Fin_Si
……….
```

- **Doble:**

DIAGRAMA DE FLUJO

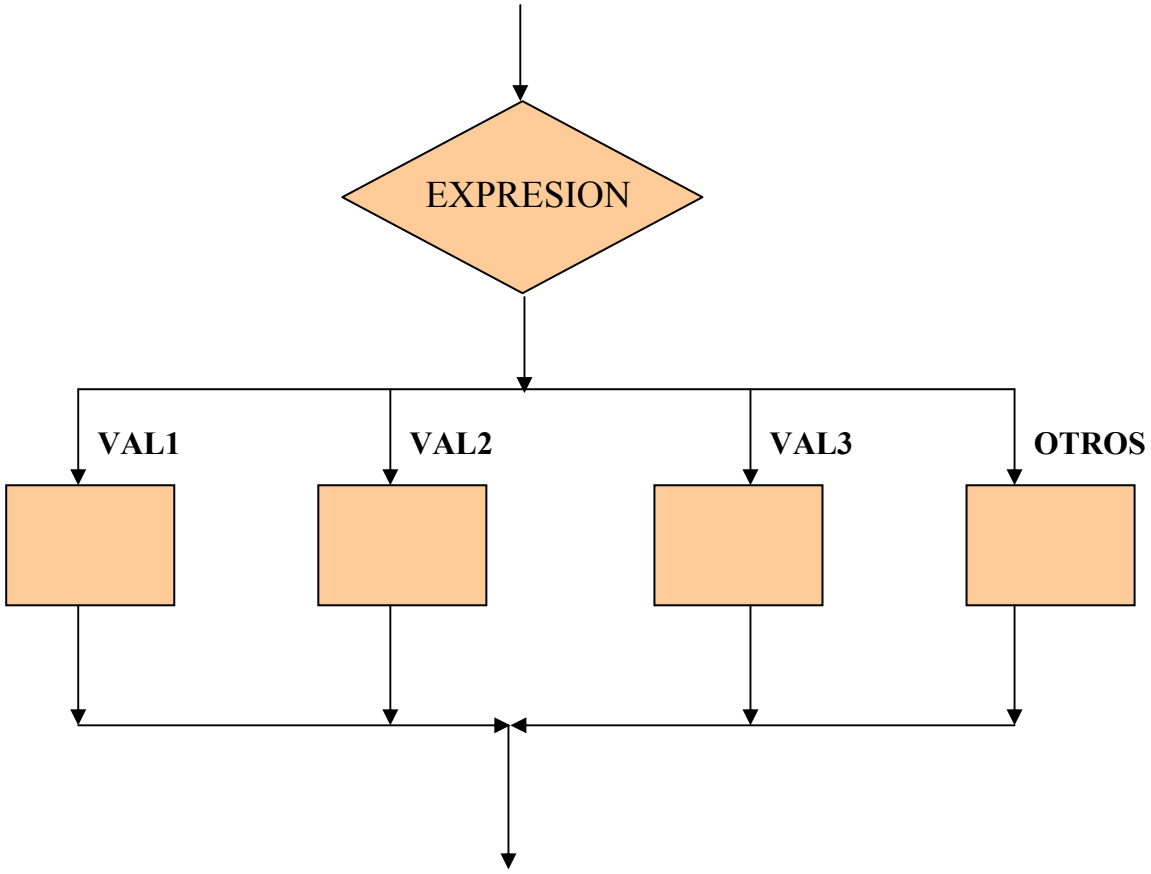


PSEUDOCÓDIGO

```
……..
Sí (condicion)
    accion1
    accion2
Sino
    accion3
    accion4
Fin_Si
……..
```

- **Múltiple:**

DIAGRAMA DE FLUJO



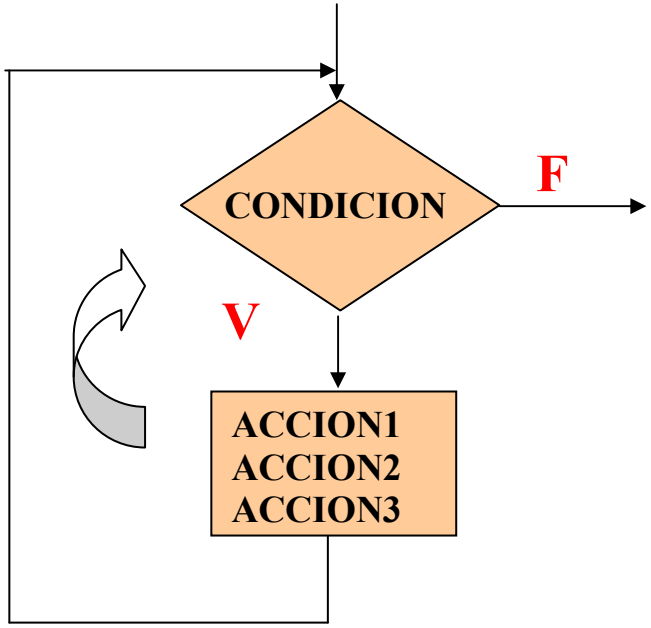
PSEUDOCÓDIGO

```
……..……..
    Según (expresión)
            Val1: accion1
            Val2: accion2
            ……..
            Otros: accionN
     Fin_Segun
……..……..
```

ESTRUCTURAS REPETITIVAS

- **Mientras:**

DIAGRAMA DE FLUJO



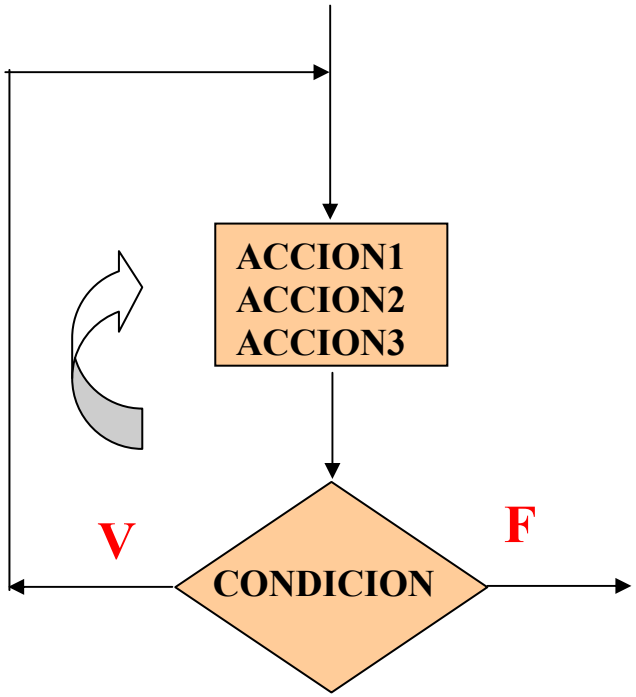
PSEUDOCÓDIGO

```
……….

Mientras (condicion)
        accion1
        accion2
        accion3
Fin_Mientras
……..
```

- **Hacer Mientras:**

DIAGRAMA DE FLUJO

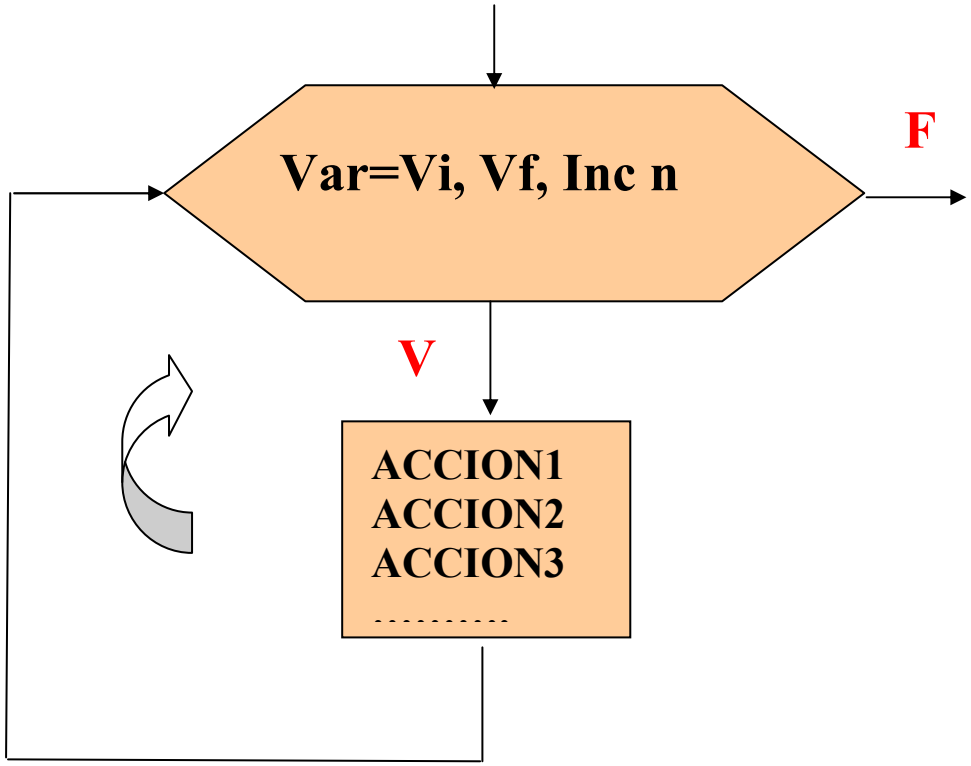


PSEUDOCÓDIGO

```
.....................
Hacer
      accion1
      accion2
      accion3
Mientras (condicion)
.....................
```

- **Para:**

DIAGRAMA DE FLUJO



PSEUDOCÓDIGO

```
……….
Para Var=Vi hasta Vf con Inc n
        accion1
        accion2
        accion3
Fin_Para
……….
```

Ejemplo1: Ingresar un número por teclado, indicar por pantalla si es positivo o negativo.

```
Programa Positivo_Negativo
Módulo: Principal
Inicio
    DATOS:
    Variables
    x: Entero
    ALGORITMO
    Leer  x
        Si (x<0)
            Escribir ('Numero negativo')
        Sino
            Escribir ('Numero positivo')
        Fin_Si
Fin
```

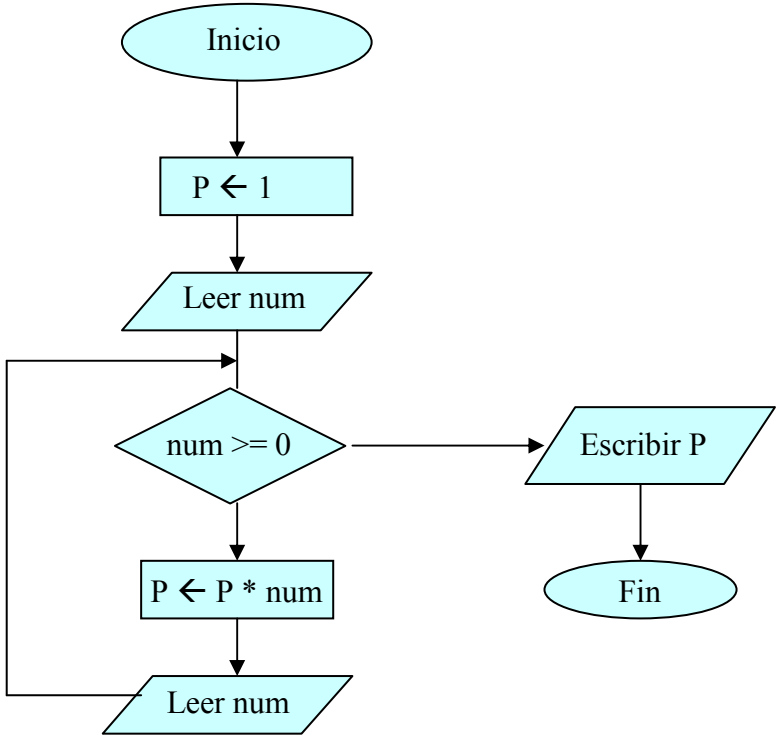
Ejemplo2: Mostrar el producto de números enteros positivos entrados por teclado hasta el ingreso de un número negativo.

Programa Producto

Módulo: Principal

```
Inicio
    DATOS:
        Variables
        P, num: entero
    ALGORITMO:
        P ← 1                  ** Si se usa el = como asignación, utilizar = = como
        Leer num               ** operador de  relación
        Mientras (num >=0)
            P ← P*num
            Leer num
        Fin_Mientras
        Escribir P
Fin
```

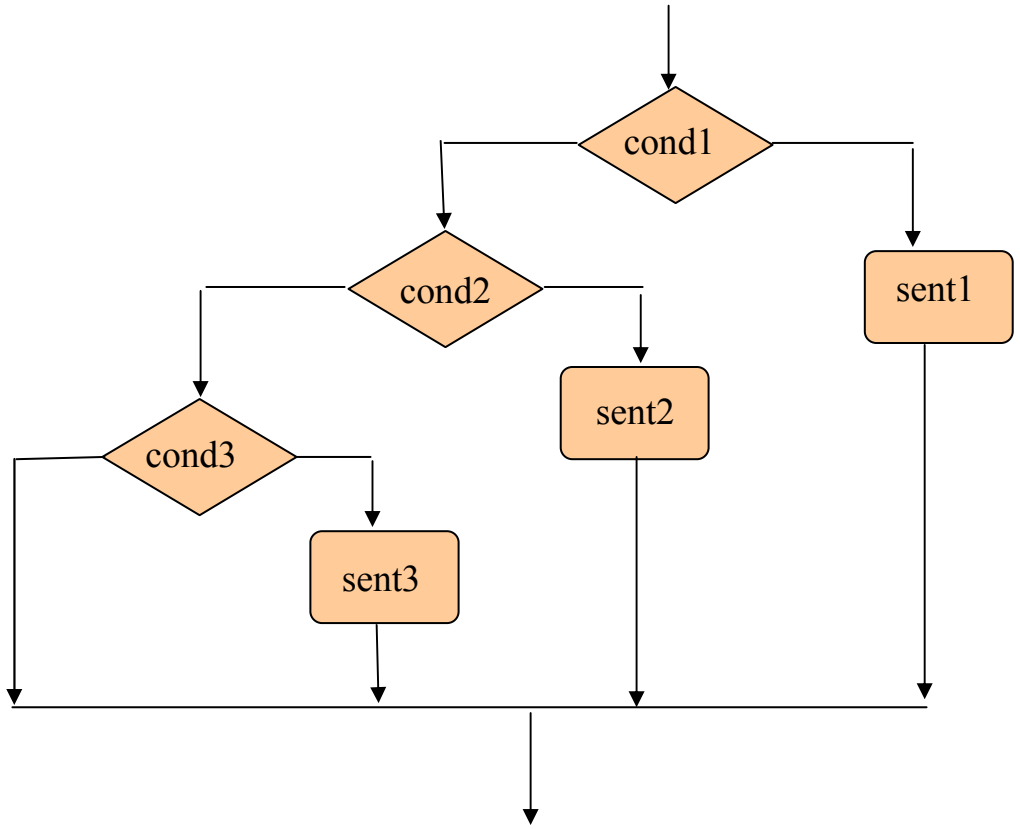
DIAGRAMA DE FLUJO DEL EJEMPLO 2:



ESTRUCTURAS ANIDADAS

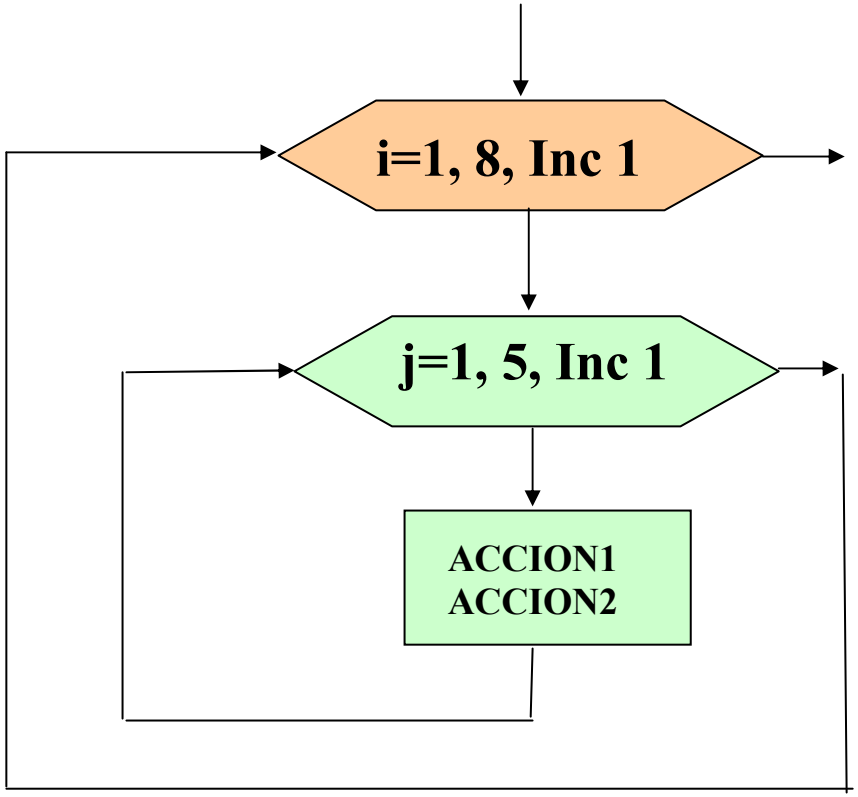
- **Anidación de condicionales:**

```
.
Si (condicion1)
    sentencia1
   Sino Si (condicion2)
          sentencia2
          Sino Si (condicion3)
                 sentencia3
                Fin_Si
          Fin_Si
Fin_Si
```



- **Anidación de bucles:**

```
Para i=1 hasta 8 con Inc 1
    Para j=1 hasta 5 con Inc 1
        accion1
        accion2
    Fin_Para
Fin _Para
```



VERIFICACIÓN DE ALGORITMOS

Diseñado el algoritmo con alguna de las herramientas mencionadas, se debe verificar y validar que funciona adecuadamente.
La prueba consiste en recorrer los distintos caminos del algoritmo, demostrando que se obtiene el resultado esperado con diferentes conjuntos de datos de entrada. Se comprueba en cada caso que la salida es la esperada. De existir un error habrá que ubicarlo y rediseñar el algoritmo.